HTML-Formatted and EDI-Formatted Data Streams: An Analysis of the Pilots

Version 1.0

06 May 1997

Contract No: 263-96-C-0300
Task Order No: ERA-0011

TYC Document ID: TYC-ERA-0011-0401.1.0

Prepared For:

Office of Policy for Extramural Research Administration
Office of Extramural Research
National Institutes of Health
Bethesda, MD

Prepared By:

TYC Associates, Inc.

TYC Associates, Inc.
12300 Twinbrook Parkway, Suite 240
Rockville, MD

TABLE OF CONTENTS

1.

Introduction

The need to exchange information between the Federal Government and the research community is critical. Traditionally, research institutions and the Federal Government have communicated by mailing preprinted business forms. Now, the Office of Policy for Extramural Research (OPERA) at the National Institutes of Health (NIH) is re-engineering the extramural grant administration process. Central to this re-engineering effort is the concept of a "Commons," which serves as an electronic mall where the grantee community can conduct business electronically with NIH. The primary technologies comprising the Commons are those technologies which enable electronic research administration via the Internet.

Two such enabling technologies are Hypertext Markup Language (HTML) and Electronic Data Interchange (EDI). HTML is a simple markup system used to create hypertext documents which are portable from one platform to another. EDI is a family of standards which specifies a common representation for business documents. HTML- and EDI-related technologies are vital to the Commons, because a key component of the Commons development is the manner and format in which users can submit data to the NIH via the Commons.

As an initial step in automating the grant administration process, TYC Associates, Inc. (TYC), in conjunction with Turner Consulting Group (TCG), developed two pilot systems. These pilots demonstrate how NIH can electronically receive select portions of the competitive grant application, store the grant application data into a relational database, and view the stored data via the Web. The differences between the pilots reside in the encoding of the grant application data. The first pilot uses the 194 transaction set [1] developed by ANSI X12 to encode the grant application data. The second pilot uses HTML-formatted data streams [2] as specified by Logistics Management Institute (LMI) to encode the grant application data. These pilots are respectively referenced as the EDI pilot and the HTML pilot within the scope of this paper.

The objective of this paper is to compare and contrast the HTML pilot (see [3] for the HTML pilot system model and requirements) withthe EDI pilot (see [4] for the EDI pilot system model and requirements). This paper is intended for readers who have knowledge of Web, EDI, and database technologies.

1.1 System References

1. *194 Grant or Assistance Application.* ANSI X12 Transaction Set. September, 1996.
2. *HTML Data Element Names for the 194 Transaction Set*. LMI Excel spreadsheet.
3. *HTML-Formatted Data Streams Pilot Requirements Document.* Version 0.8. TYC Associates, Inc. Feb. 28, 1997. TYC Document ID TYC-ERA-0011-0101.08.
4. *Requirements for the NIH EDI Prototype System.* Version 1. TYC Associates, Inc. July 8, 1996. TYC Document ID TYC-ERA-0003-0201.1.

5. *Maintaining Referential Integrity Within a Relational Database When Mapping the X12 194 Transaction Set.* Version 1.0. TYC Associates, Inc. Jan. 17, 1997. TYC Document ID TYC-ERA-0003-0801.1.0.

1.2 Terms and Abbreviations

ANSI	American National Standards Institute
CGI	Common Gateway Interface
EDI	Electronic Data Interchange
FTE	Full Time Employee
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LMI	Logistics Management Institute
NIH	National Institutes of Health
ODBC	Open Data Base Connectivity
SQL	Structured Query Language
OPERA	Office of Policy for Extramural Research

1.3 Organization of Document

This document, "HTML-Formatted and EDI-Formatted Data Streams: An Analysis of the Pilots", contains three major sections. Section 1 introduces the paper. Section 2 analyzes the HTML and EDI pilot systems, and section 3 presents recommendations based on the analysis.

2. Pilot Analysis

TYC has implemented both an HTML pilot and an EDI pilot for NIH. These pilots are functionally equivalent in that they both enable NIH to:

1. receive competitive grant application data,
2. store the data into a relational database,
3. and view the data via a Web browser.

Both pilots have successfully fulfilled all requirements pertaining to these functional areas, as defined by the requirements documents for the respective systems [3,4].

This section presents an analysis of the two systems based on the design and implementation of the pilots. The analysis comprises integration issues, availability (in terms of products and an encoding syntax), and cost.

Unless stated otherwise, all issues represent the perspective of NIH as an awarding Federal agency. Issues pertaining to grantee organizations are stated as such.

2.1 Integration Issues

In order to receive grant application data and load this data into a database, the HTML and EDI pilot systems integrate a communications component and a database component. Issues relevant to the integration of these components are described below.

Both the HTML and EDI pilot systems support a file-based model. This means that a grantee organization can store HTML-formatted data or EDI data in a file and upload (i.e., HTTP file upload) the file to the appropriate NIH server. The main issue here is that the difference between an HTML-formatted data stream and an EDI data stream is the syntax of the stream itself; it has no relevance to the communications medium. Thus, there is no difference between the HTML and EDI pilot systems in how the data is communicated with NIH.

The competitive grant application data received by NIH is stored in a relational database. The data is loaded into the database using the following methods.

1. **HTML Pilot System:** The receipt of HTML-formatted data triggers a Common Gateway Interface (CGI) script which parses the data, and generates the Structured Query Language (SQL) commands to load the grant application data into the database.

2. **EDI Pilot System:** The receipt of EDI data triggers the EDI translator which parses the data. The EDI translator is integrated with an Open Database Connectivity (ODBC) driver which generates the SQL commands to load the grant application data into the database.

The primary issue with database integration is initial programming verses flexibility. The HTML pilot system requires more initial programming than the EDI pilot system to parse the data, generate the SQL commands, and load the data into the database. Initial coding for the EDI system is limited to configuring the data map for the translator (i.e., parsing the data) and configuring the ODBC driver (i.e., loading the data into the database). Configuring the EDI system is not a trivial matter; however, an EDI translator can provide visual tools for data parsing and loading.

Within the EDI pilot system, EDI data is written directly to the relational database via the ODBC driver. This approach highlighted a lack of flexibility within the EDI system. The lack of flexibility is not due to the functionality provided by the translator or the ODBC driver; but, rather, by the denormalized encoding of the grant application data (i.e., the X12 194 transaction set). A major problem with loading EDI data into a relational database is how to maintain referential integrity among database tables [5]; i.e., generating

and coordinating the primary and foreign key values amongst the various tables involved in the data capture. This problem resulted in TYC using *temporary* tables for storing the denormalized 194 data, and using database triggers to generate key values and move the data from the temporary tables to their *final* destination tables. The use of temporary tables and database triggers is not required by the HTML pilot system.

The EDI system could have been modeled differently. The grant application data could have been written to a flat-file, which is the traditional EDI approach. A program could then read and parse the file, and write the grant application data to the database (similar to the CGI script for the HTML system). Temporary tables and database triggers would no longer be needed within the EDI system. The problem with this solution, however, is that it reduces the functionality of the EDI translator to a parser. In this case, the EDI translator would only enact the first pass of a two-pass parser, with the second pass performed by the program that accesses the flat-file and writes the grant application data to the database. This two-pass approach adds a level of redundancy to the architecture.

From the perspective of a grantee organization, the issue of referential integrity might not be applicable. NIH needs to address referential integrity because grant application data is converted from a denormalized syntax (i.e., the X12 194) to a normalized format (i.e., tables within a relational database). Grantee organizations, when submitting grant applications, are faced with the inverse scenario - converting normalized data to a denormalized syntax. This process might be a simple matter; however, more research needs to be performed to understand all the relevant issues.

One final issue relating to integration is that of *forms*. The HTML-formatted data streams used to communicate grant application data can be linked to Web forms. This means that a grantee organization can create a Web form for entering grant application data, and the data stream produced from *submitting* the form can be made consistent with the syntax required by NIH (or other grant awarding agency). Thus, the HTML-formatted data stream can be produced automatically by a Web browser.

2.2 Availability

There are many commercial EDI translators available today. Although few support ODBC technology, the NIH EDI system is implemented using commercial products only. This is not to imply that integrating an EDI translator with a relational database is a simple exercise of plug-and-play. There is a non-trivial amount of configuration required for the integration. In a sense, one can compare the amount of configuration required for the EDI pilot against the amount of initial programming required for the HTML pilot; the effort needed for the EDI pilot is less, but not substantially less.

There are no commercial products for implementing the HTML system. The concept is new, and requires some amount of coding to implement. The coding is limited to parsing the grant application data and generating the SQL commands needed to load the database.

There are commercially- and publicly-available tools to facilitate the implementation of an HTML system.

The novelty of using HTML-formatted data streams to communicate competitive grant application data highlights its greatest disadvantage: the syntax is not defined. There is no commonly accepted, tested syntax for conveying grant application data via an HTML-formatted data stream. LMI has taken the initiative to provide a first draft of data elements and their attributes (e.g., name, length, whether there can be multiple occurrences, etc.) [2], however, the draft syntax must be tested. In contrast, the ANSI X12 committee has developed the 194 transaction set which specifies the encoding for competitive grant application data. Both the 194 transaction set and the set of LMI data elements require an *Implementation Guide* before grantee organizations can meaningful communicate grant application data to NIH.

The novelty of the HTML-formatted data stream concept and the lack of a tested, accepted encoding syntax, however, underlies a benefit. Problems have been identified with the X12 194 syntax in that it more closely mirrors a paper form than a normalized data model. Whereas substantial changes to the 194 syntax are difficult (if not impossible)[1] changing the presently-defined HTML-formatted data stream syntax is less complicated. The encoding of HTML-formatted data streams can be made generic, but also (to a greater degree than the 194) normalized.

2.3 Cost

The costs of implementing and maintaining an EDI system can be categorized as follows (note that all costs described in this section pertain to grantee organizations as well as to NIH):

Software: costs pertaining to the purchase of EDI software and ODBC driver (ranging from 8K to 90K)
Fees: costs pertaining to annual fees, which include maintenance, software upgrades, new releases of EDI standards, and support.
FTEs: costs pertaining to the salaries of those who maintain the EDI software

The HTML system, because it is not (currently) based on commercial products, does not have the software and fees associated with the EDI system. The cost to implement an HTML system is measured in the number ofFTEs required to parse (for NIH) or create (for a grantee organization) the HTML-formatted data stream.

[1] The X12 194 transaction set is developed and maintained by the X12 committee. Any changes to the 194 must be brought before, agreed to, and enacted by the committee. Given the diverse requirements of committee members, and the time required to implement changes (e.g., changes require multi-layer voting and approval), it is unlikely that substantial changes to the 194 can be effected.

In terms of pilot maintenance, the number ofFTEs needed to maintain the HTML pilot is comparable to that of the EDI pilot.

One final issue pertaining to maintenance is environment. Grantee organizations are, by nature, research oriented, and will most likely have a staff experienced with CGI and HTML technologies. This might not be true of EDI technology.

3. Recommendations

Section 2 presented a comparison of the HTML and EDI pilot systems developed by TYC for NIH. This section presents some conclusions and recommendations based on that comparison.

This paper makes no attempt to answer the question of which system is*better* for encoding competitive grant application data. Federal agencies and grantee organizations have distinct and diverse requirements. Clearly, different requirements will warrant different solutions.

The fundamental questions that many will ask regarding these systems are:

1. What is the cost to procure the software needed implement and maintain the systems?
2. What level of effort is required to implement and maintain the systems?

These questions are addressed below.

The cost of software needed to implement and maintain an EDI system comprises the initial purchase of the EDI software and the annual EDI software fees. This is true for either a grantee organization or an awarding Federal agency. In contrast, there are *no software costs* associated with implementing and maintaining an HTML system.

Level of effort is more difficult to describe than procurement costs. For a grantee organization, implementing an EDI system comprises installing, configuring, and integrating the EDI software. Implementing an HTML system comprises whatever is required to generate the HTML-formatted data stream. If linked to Web forms, the data stream can be generated automatically by a Web browser; otherwise, coding is required to create the data stream. Comparatively speaking, the initial level of effort required by a grantee organization to implement an HTML system is greater than the initial effort required to implement an EDI system, but not substantially greater. Once implemented, each system requires minimal maintenance.

For an awarding Federal agency, implementing an HTML system involves parsing, validating, and storing (in a database or in files) the grant application data. Implementing an EDI system comprises installing, configuring, and integrating the EDI software. With the EDI system, however, the awarding agency must contend with the issue of referential

integrity if writing grant application data directly to a database. In this scenario, the level of effort required for implementing the EDI system actually becomes greater than that required for implementing the HTML system. Once implemented, each system requires minimal maintenance.

These conclusions, thus far, have not addressed the strength of EDI. This strength is that EDI is generic. X12 provides the encoding syntax for procurement transactions, financial transactions, grant application transactions, etc. If EDI is already employed within an organization, using the X12 194 transaction set to convey grant application data becomes a reasonable solution. The EDI translator used for procurement functions, for example, can also be used for grant applications. Thus, there are no initial procurement costs, because the EDI software is already in-house.

If EDI is not used internal to an organization, the value of using the established encoding syntax offered by EDI must be weighed against the investment required for EDI. There are (possibly substantial) initial and maintenance costs associated with the deployment of EDI, as well as a non-trivial amount of initial configuration. In these scenarios, the use of HTML-formatted data streams offers an attractive, cost-effective alternative to EDI.